Uni-Bot (ESP32)
Calibration

Monitor+

Ptch SafeMode OFF
PtchGyrAng 0.31°
SbPtchSp 0.00°   0.025
PtchErr 0.00
Ptch_Op 0.00   0.00

COM: COM5   Rx: -   Tx: ~   TechKnowTone
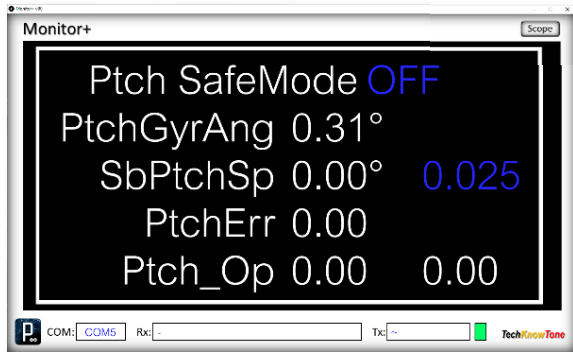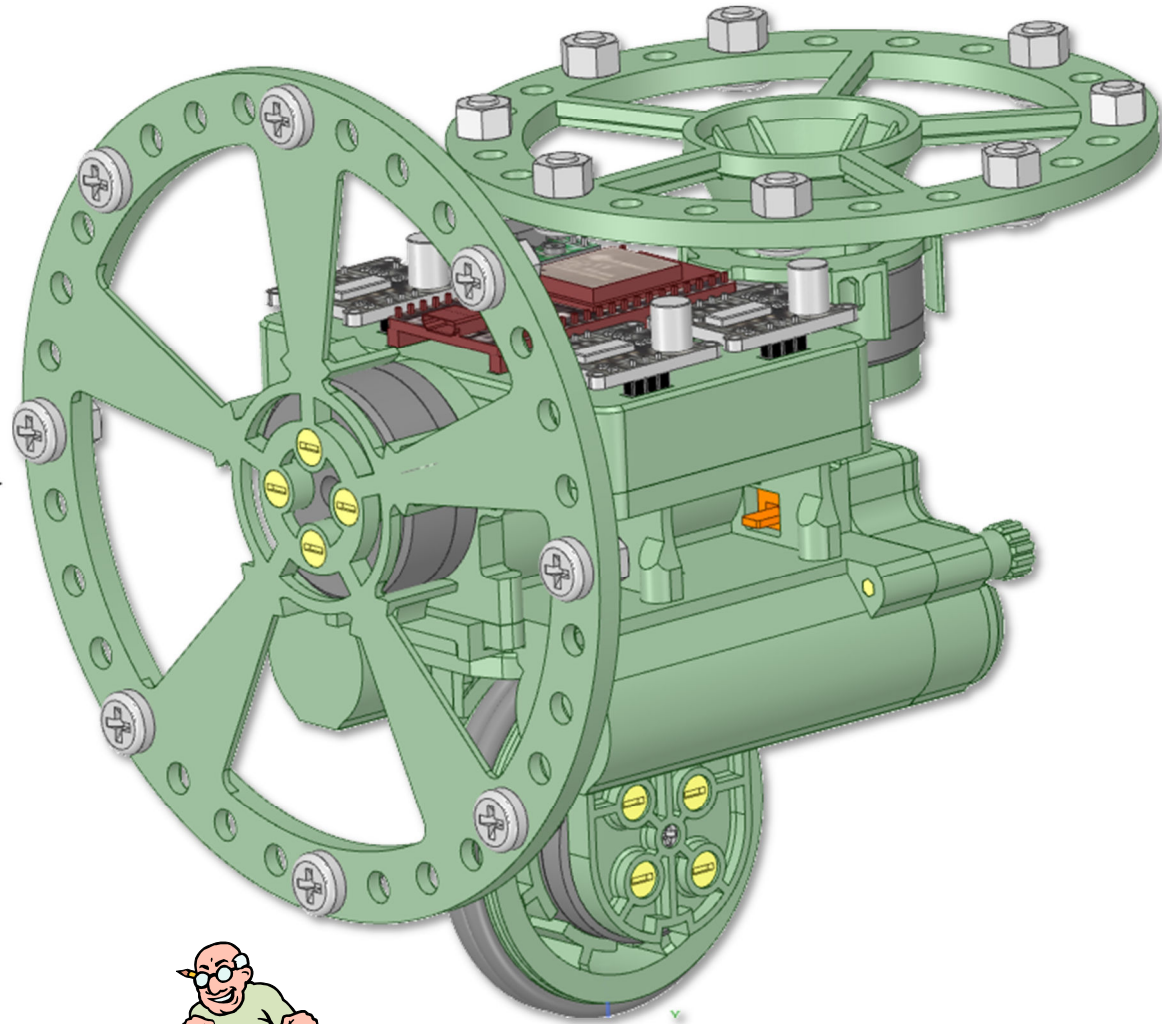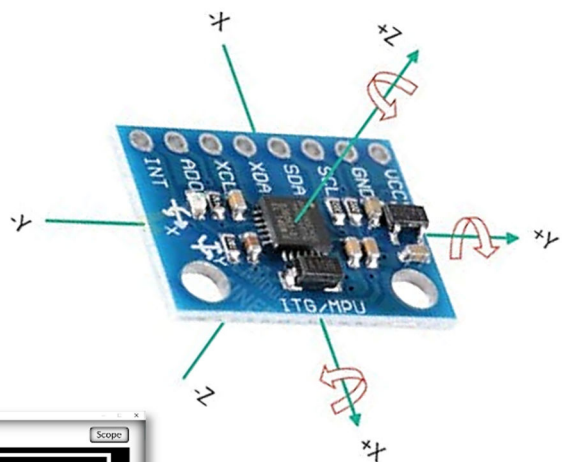
Calibration is an essential process!

Voltage Regulator

Set up the voltage regulator independently using an external power supply and multimeter, <u>before</u> you plug in the micro and drivers. The input voltage source should be > 8 volts, and we trim the regulators output to achieve 3.3 volts. This is normally done by turning the small potentiometer on the pcb in a clock-wise direction, to reduce the output voltage.

Once the robot is wired and assembled, plug in the voltage regulator, ensuring the correct orientation, with the potentiometer and inductor towards the lower edge. Install the three batteries, and check that the output of the voltage regulator is at 3.3 volts.

Now insert the ESP32 micro, with the power off, and download the code into it. When power is re-applied, this should confirm that the RGB LED's are working. All three SimpleFOC mini drivers power LEDs should also light up.

The accuracy of the analogue to digital converter (ADC) in the ESP32 can be improved by calibrating it's measurements against a multimeter, whilst using an adjustable voltage supply. This is important for setting the correct motor PWM values, to prevent them over-heating.

I soldered a 1MΩ resistor across the voltage regulators input pins, to make it easier to attach an external power supply for these measurements, and obtained the following values to enter into the code:

```
10.0v == #define BatCritical 2895
11.4v == #define BatHigh 3402
12.3v == #define BatMax 3841
         #define BatPin 36
10.8v == #define BatWarn 3175
```
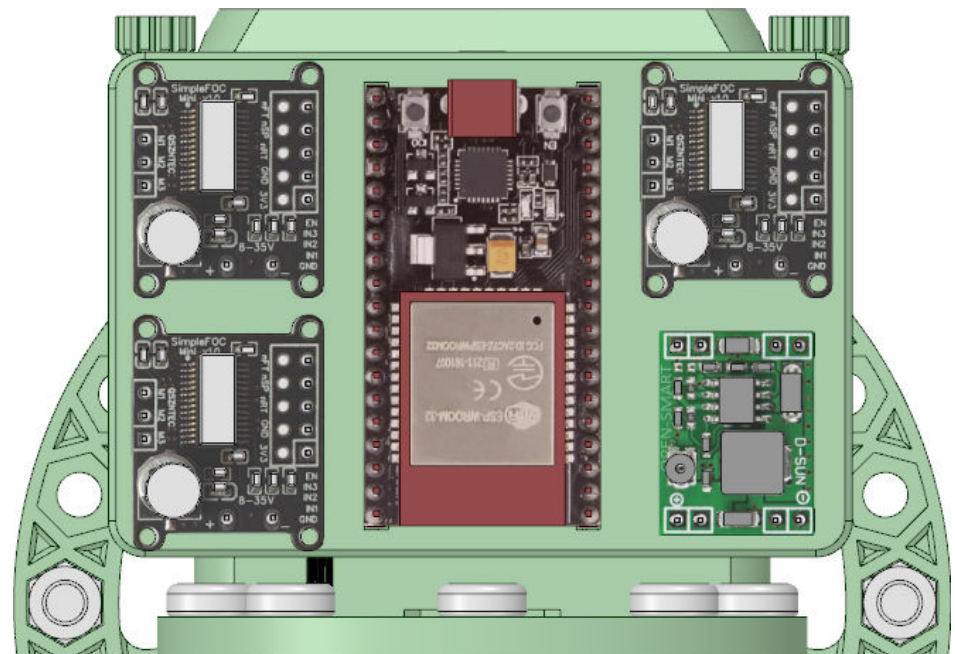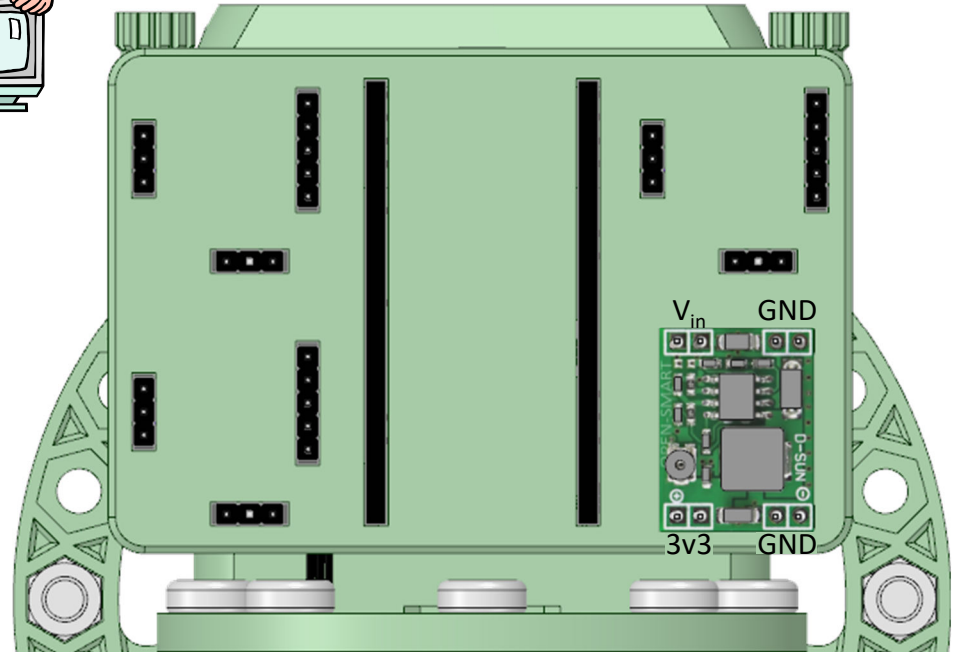
MPU6050A Orientation

With the MPU6050 mounted on the right leg as shown, the following applies:

Pitch - Z gyro, -ve tilt forwards, +ve tilt backwards
- Y accelerometer, -ve lean forwards, +ve lean backwards
Roll - Y gyro, +ve tilt right, -ve tilt left
- Z accelerometer, -ve lean right, +ve lean left
Yaw - X gyro, +ve turning right ,-ve turning left
- X accelerometer, , -ve upright, +ve upside down

Note that the X-axis accelerometer is pointing downwards, when the robot is upright and balancing. Hence it is greatly influenced by gravity and give a max negative reading.

The X-axis value is negative because the gravitational force is acting on the sensor as if it were accelerating upwards, and therefore in the opposite direction to the polarity indicated by the diagram.

The MPU6050 sensor values will contain offsets, due to manufacturing tolerances, and may not give maximum values. We therefore need to determine these offsets, and then remove them in our code, so as to improve the overall accuracy of the system.

The next page explains how you do this.

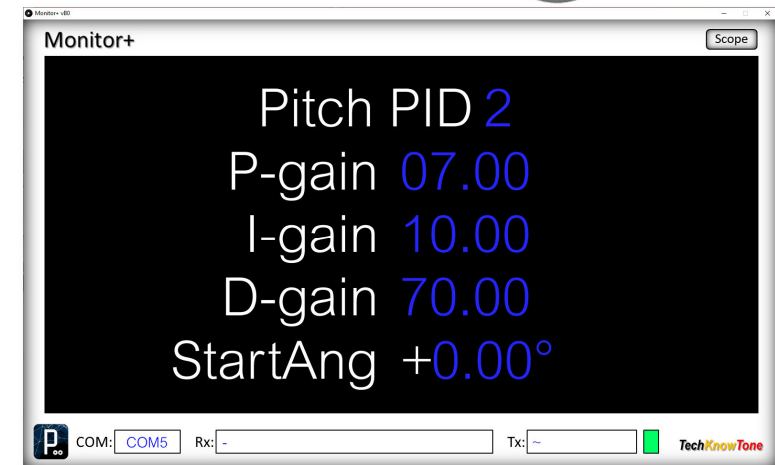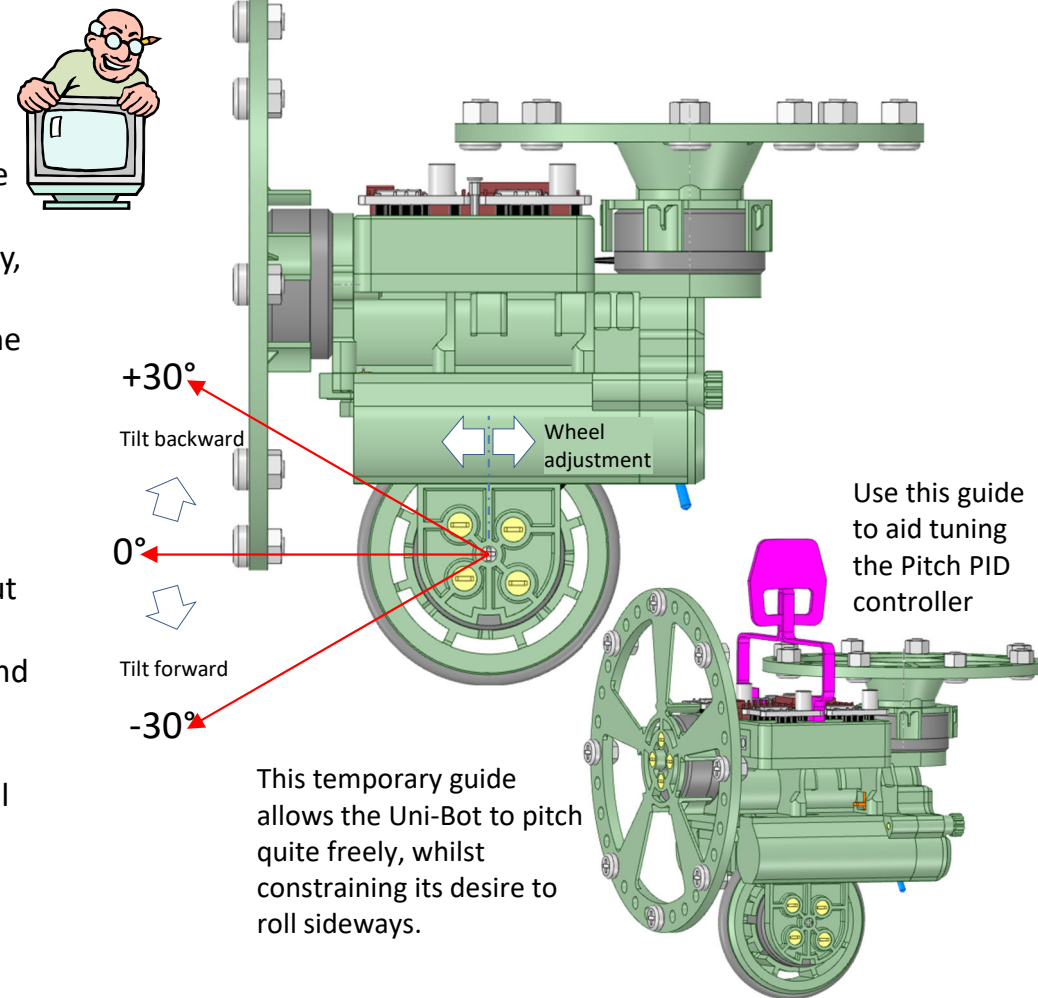**MPU6050A**
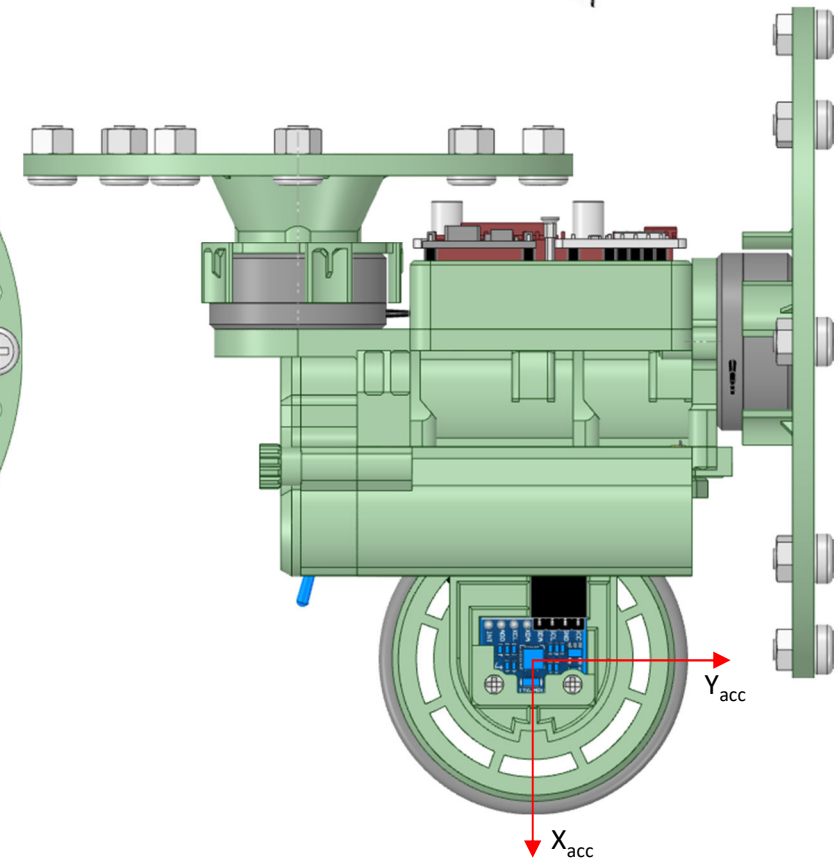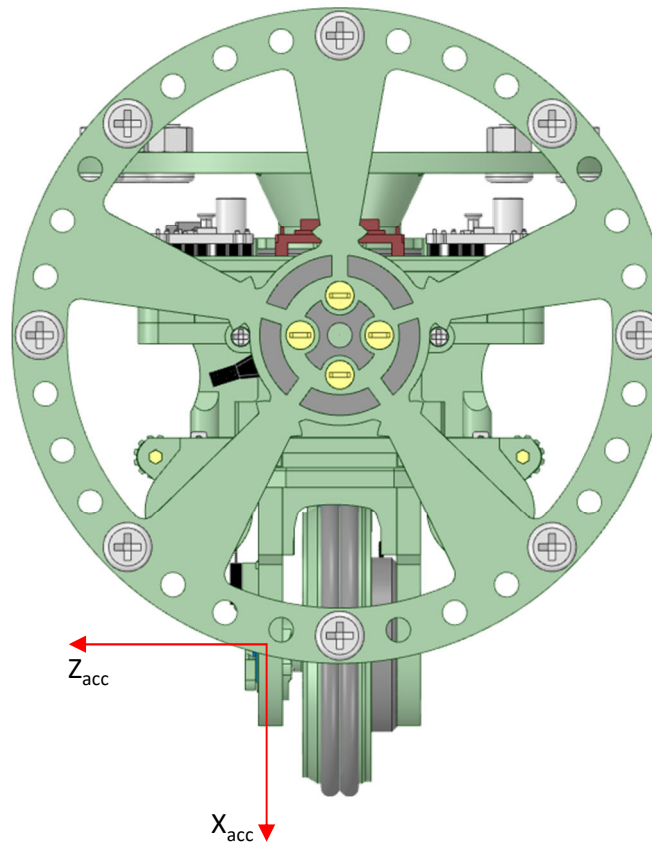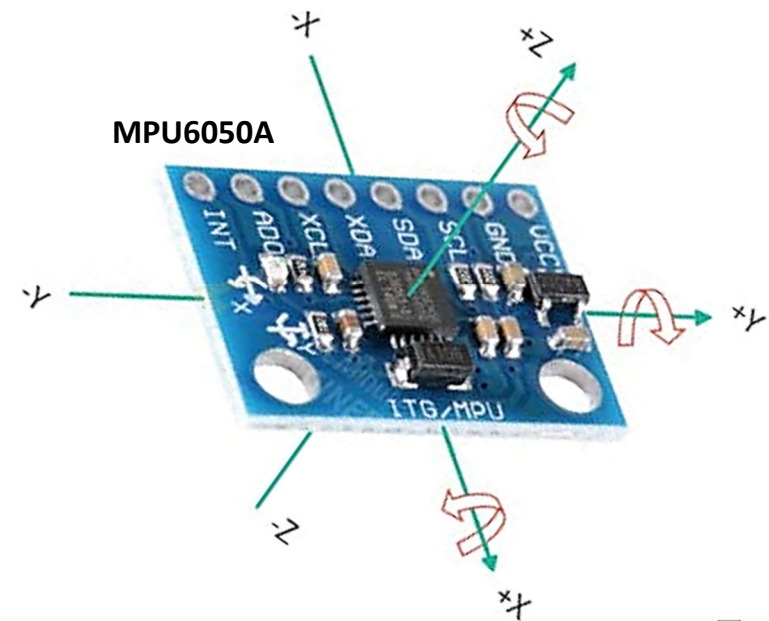
Uni-Bot Pitch PID Adjustment

The aim is to balance the Uni-Bot using the 'Pitch' PID controller code, when the robot is level, at 0° facing forwards. The drive wheel is mounted on an adjustable slide to make this possible; however, the position of the drive wheel will depend on what weights you have fitted to the roll and turn reaction wheels, respectively, and the distribution of weight within the robot. So the angle at which the robot can be balanced my not be 0° initially, and we use the self-balancing nature of the code to find the set-point for balance; a code variable named SbPtchSp.

If the pitch start angle, PtchStartAng, is set to 0° initially, then this is the starting point for the auto-adjustment of SbPtchSp. If this isn't the point of balance, the robot will naturally fall forwards or backwards, developing an error signal in the PID controller, which in turn will move the drive wheel to compensate for this out of balance state. The code senses this condition, and automatically adjusts the balance setpoint SbPtchSp to counter this behaviour. The robot will drive back and forth, for a few moments initially, until SbPtchSp is at the correct value.

To start with, the position of the drive wheel may be such that the robots natural pitch balance is out by several degrees, and the auto-adjustment code can't compensate for this. If this is the case the robot will have a tendency to run off, forwards or backwards, depending on the error. We can fix this, by adjusting the start angle, giving the code a greater chance of achieving balance.

If the robot wants to drive backwards, then adjust the start angle to a negative value, say -2.00° and try it again. And if the robot wants to drive forwards at the start, then make the start angle a positive value, like +2.00°.

Through a process of trial and error we are adjusting the position of the drive wheel, such that the start angle can be set to 0°, and the automatic self-balance set point SbPtchSp adjustment ends up very close to that value.

+30°

Tilt backward

0°

Tilt forward

-30°

Wheel adjustment

Use this guide to aid tuning the Pitch PID controller

This temporary guide allows the Uni-Bot to pitch quite freely, whilst constraining its desire to roll sideways.

Pitch PID 2
P-gain 07.00
I-gain 10.00
D-gain 70.00
StartAng +0.00°

Monitor+ v80
Monitor+   Scope

P   COM: COM5   Rx: -   Tx: ~   TechKnowTone

∞ + ESP32 Page 4

Uni-Bot Roll PID Adjustment

Tune the Pitch PID controller, using the 'Guide' attachment before attempting to tune the Roll PID controller. As this will give you experience of tuning a controller before attempting the greater challenge. As with the Pitch PID controller, you can adjust the Roll PID coefficients simply by clicking on the blue digits in the Monitor+ app. Any changes will be lost when the micro is reset, so you note them down and make changes to the code, once you have values that you want to use.

The Roll PID controller is very sensitive to being given the correct start angle, which is the angle at which the controller becomes active during the initiation process (see video). The self-balancing adjustment will make for some allowance in this initially, so you observe the angle to which the system is trying to get to and set that as the start angle. You will see from the Monitor+ screen shot below, that my Uni-Bot Roll PID self-balances at -1.00° and that is my start angle in code.

Using GPn = 4 in the 'scope' display, enables you to view what is happening within the Roll PID controller whilst the Uni-Bot is self-balancing. You should expect to see the motor velocity changing in in a sinusoidal fashion, as the robot sways from side to side about the balance point; with the P-gain term dominating the output, whilst the D-gain provides corrective action, and the I-gain provides an underlying sinusoidal trace in response to small changes in angular error.

If you use the same arrangement of reaction wheel weights as me, you should come up with very similar Roll PID coefficients.

-ve Roll angle   +ve Roll angle
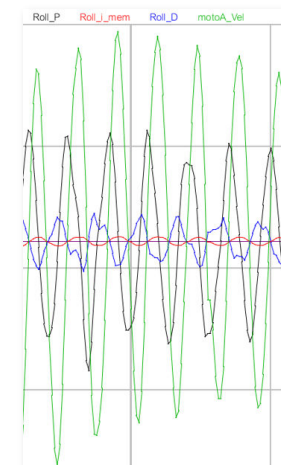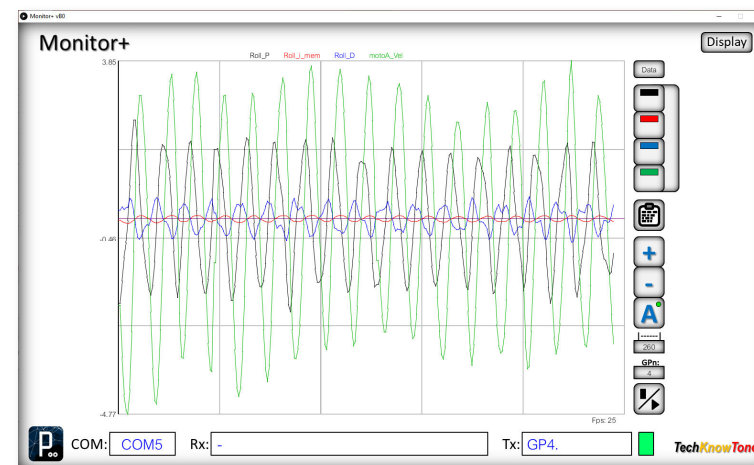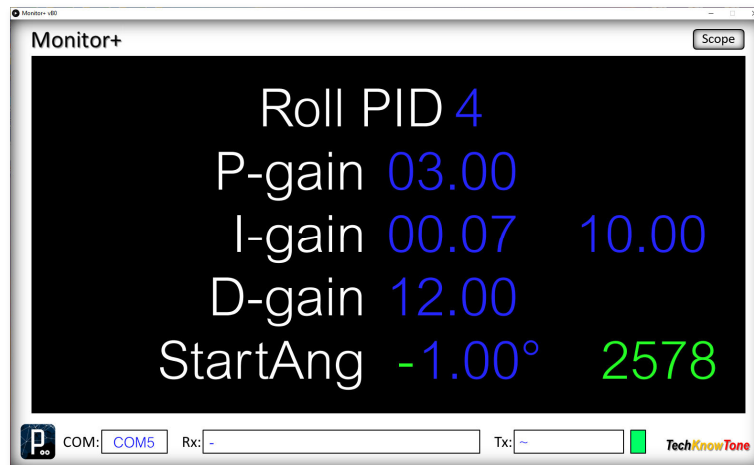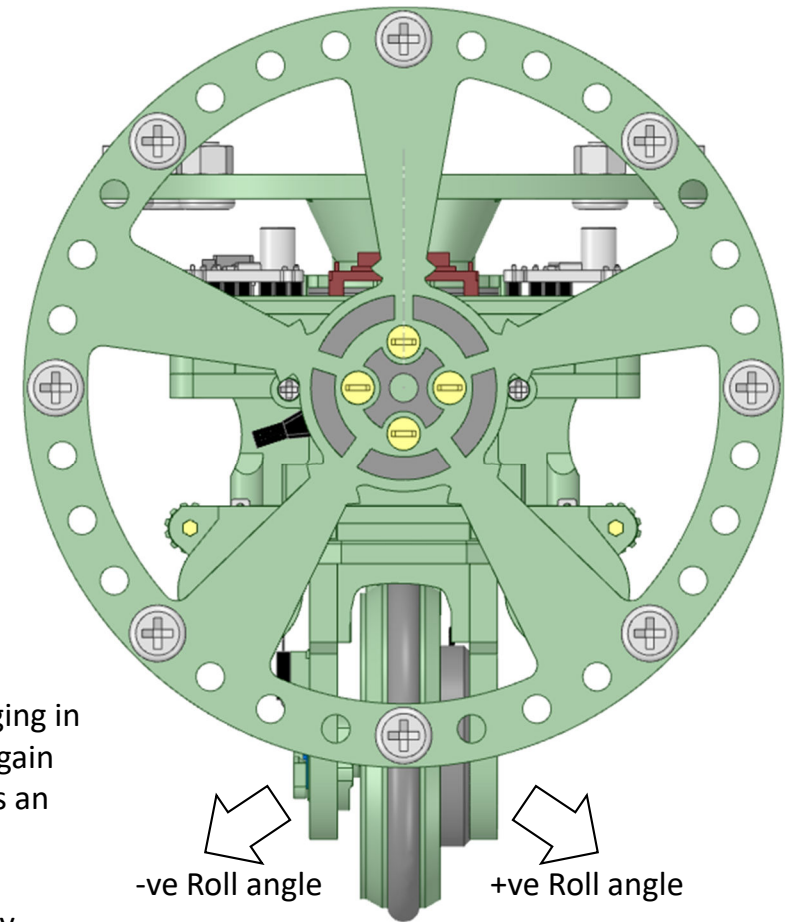
Uni-Bot Turn Speed Adjustment

We know that the turning force exhibited by a reaction wheel is proportional to its rate of acceleration. To increase the turning force developed, we either add weights to the wheel or increase its rate of acceleration. But adding too much weight will affect the robots ability to balance, and its pitch self-balance setpoint. And we can't just keep accelerating the wheel, due to max speed limitations of the BLDC motor and the SimpleFOC driver. So there is a compromise to be had here.

So I chose to use eight M5 nuts and bolts on the turn wheel, as acceleration test suggested that would be more than enough to turn the robot round. With those weights added, I needed to determine a practical max. rate of acceleration to achieve suitable turning, and also a minimum rate of deceleration to apply when slowing the wheel down.

The 'Turn Control' screen was coded to achieve this. Using this display you can click on the blue digits to set values for max acceleration and min deceleration, whilst using the Wii Nunchuk controller to trigger the demand.

The aim is not to have the Uni-Bot turning like a spinning top, when a demand is applied, and for the wheel to slow down relatively quickly, so that further demands can be applied.

The Monitor+ screen shot here shows the values which I came up with, for my Uni-Bot. You may like this, or determine a more/less aggressive turning characteristic for your robot.

The ratio of Max/Min appears to be in the order of 4:1
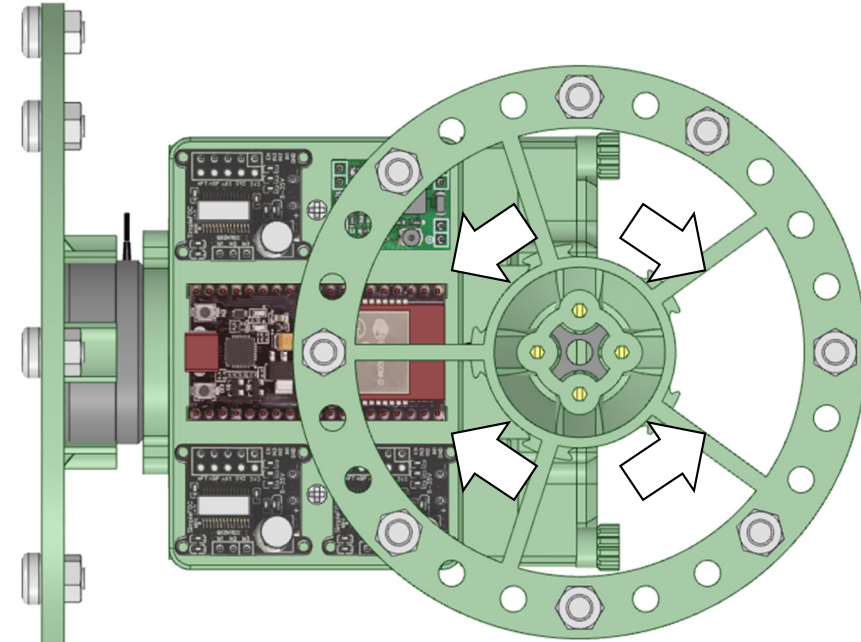
If the turn wheel accelerates clockwise, then turning force on Uni-Bot is anti-clockwise.
If the turn wheel accelerates anti-clockwise, then turning force on Uni-Bot is clockwise.

Monitor+ [Scope]

Turn Control

JoyX 128
Min Acc 0.060
Max Acc 0.250
Turn 11.9°

COM: COM5   Rx: -   Tx: ~   TechKnowTone